WheeledWalks App

David Lee – 11055579 – [david@lee42.com](mailto:david@lee42.com)

Developed for Mobile Applications Development 31285 at UTS

An Android Application for Data Logging of wheelchair accessible bushwalks

Table of Contents

[Table of Figures 3](#_Toc434234764)

[1 Introduction 4](#_Toc434234765)

[1.1 Application Vision 4](#_Toc434234766)

[1.2 Scope 4](#_Toc434234767)

[1.3 Document Version History 5](#_Toc434234768)

[2 Functionality Overview 5](#_Toc434234769)

[2.1 Feature Summary 6](#_Toc434234770)

[2.1.1 Walk Card display and simple walk database 6](#_Toc434234771)

[2.1.2 Editing, Deleting, and Review of recorded walks 6](#_Toc434234772)

[2.1.3 Add new walk 6](#_Toc434234773)

[2.1.4 Google maps view 6](#_Toc434234774)

[2.1.5 Step and obstacle map markers 6](#_Toc434234775)

[2.1.6 Photo option for obstacle markers 6](#_Toc434234776)

[2.1.7 Logging of GPS, Accelerometer, and marker data 6](#_Toc434234777)

[2.1.8 Adding descriptions, rating, and thumbnail images to walks 6](#_Toc434234778)

[2.1.9 Minimal data processing for display purposes 7](#_Toc434234779)

[2.2 Application Flow Diagram 7](#_Toc434234780)

[3 Design 8](#_Toc434234781)

[3.1 Overall Components overview 8](#_Toc434234782)

[3.2 Mock-ups 9](#_Toc434234783)

[3.3 Used Application Programming Interfaces 13](#_Toc434234784)

[3.3.1 Google Maps API 13](#_Toc434234785)

[3.3.2 Speech to Text API 13](#_Toc434234786)

[3.3.3 GPS Services 13](#_Toc434234787)

[3.3.4 Sensor Services 13](#_Toc434234788)

[3.3.5 Gallery and Camera 13](#_Toc434234789)

[4 Application Structure 13](#_Toc434234790)

[4.1 Fragments 13](#_Toc434234791)

[4.1.1 AddObstacleDialogFragment 13](#_Toc434234792)

[4.1.2 AddStepDialogFragment 14](#_Toc434234793)

[4.1.3 CustomWalkMapFragment 14](#_Toc434234794)

[4.1.4 EditWalkDialogFragment 14](#_Toc434234795)

[4.1.5 RawDataFragment 14](#_Toc434234796)

[4.1.6 SearchWalksDialogFragment 14](#_Toc434234797)

[4.1.7 StatsFragment 14](#_Toc434234798)

[4.1.8 WalkMapFragment 14](#_Toc434234799)

[4.1.9 LogFileViewerDialog 14](#_Toc434234800)

[4.2 Activities 14](#_Toc434234801)

[4.2.1 StartScreenActivity 14](#_Toc434234802)

[4.2.2 NewWalkActivity 14](#_Toc434234803)

[4.2.3 RecordWalkActivity 15](#_Toc434234804)

[4.2.4 FinaliseWalkActivity 15](#_Toc434234805)

[4.2.5 SettingsActivity 15](#_Toc434234806)

[4.3 Services 15](#_Toc434234807)

[4.3.1 AccelerometerManager 15](#_Toc434234808)

[4.3.2 GpsTracker 15](#_Toc434234809)

[4.3.3 LogWriter 15](#_Toc434234810)

[4.4 Models 15](#_Toc434234811)

[4.4.1 AccelerometerData 15](#_Toc434234812)

[4.4.2 Walk 15](#_Toc434234813)

[4.4.3 Platform 15](#_Toc434234814)

[4.4.4 WalkDbConstants 15](#_Toc434234815)

[4.4.5 Constants 15](#_Toc434234816)

[4.5 Controllers 15](#_Toc434234817)

[4.5.1 NavDrawerAdapter 15](#_Toc434234818)

[4.5.2 WalkCardViewAdapter 15](#_Toc434234819)

[4.5.3 WalkCardViewHolder 16](#_Toc434234820)

[4.5.4 WalksDatabaseHelper 16](#_Toc434234821)

[4.6 Utilities 16](#_Toc434234822)

[4.6.1 Utils 16](#_Toc434234823)

[5 Data Structure 16](#_Toc434234824)

[6 Testing 16](#_Toc434234825)

[7 Conclusion 17](#_Toc434234826)

[8 References 18](#_Toc434234827)

# Table of Figures

[Figure 1 - Application Flow Diagram 7](#_Toc434234828)

[Figure 2 - Start Screen 9](#_Toc434234829)

[Figure 3 - Navigation Drawer 9](#_Toc434234830)

[Figure 4 - Settings 9](#_Toc434234831)

[Figure 5 - New walk Prompt 9](#_Toc434234832)

[Figure 6 - New Walk Activity 10](#_Toc434234833)

[Figure 7 - Record Walk Raw Data Tab 10](#_Toc434234834)

[Figure 8 - Record Walk Map Tab 10](#_Toc434234835)

[Figure 9 - Record Walk Add Step Dialog 10](#_Toc434234836)

[Figure 10 - Record Walk Add Obstacle Dialog 11](#_Toc434234837)

[Figure 11 - Map Now Showing Markers 11](#_Toc434234838)

[Figure 12 - Record Walk Stats Tab 11](#_Toc434234839)

[Figure 13 - Finalise Walk Warning 11](#_Toc434234840)

[Figure 14 - Finalise Walk Activity 12](#_Toc434234841)

[Figure 15 - Walk Added To Start Page 12](#_Toc434234842)

[Figure 16 - Edit Walk Dialog 12](#_Toc434234843)

[Figure 17 - View Log Dialog 12](#_Toc434234844)

[Figure 18 - Filtered Log View 13](#_Toc434234845)

[Figure 19 - Delete Walk Prompt 13](#_Toc434234846)

# Introduction

The NSW National Parks and Wildlife service currently lists only 46 walks in NSW as being “access friendly”, otherwise known as wheelchair accessible. Almost all of these walks are less than 2km long, usually on sealed paths, so more adventurous wheelchair users find it hard to get information on the many longer yet still wheelchair accessible paths that exist across NSW. This is due to there being no formal classification as to what “wheelchair Accessible” is, and no objective grading system for these walks. I have been approached by Wildwalks.com.au and my paraplegic friend Helen to help them develop a database of wheelchair friendly walks across Australia, with a grading system and track notes to be displayed on the Wildwalks website.

The App I have developed takes advantage of many features of the modern smartphone. The phone can be rigidly mounted to a wheelchair, and then taken on bushwalks. The app tracks and periodically logs the GPS location of the phone to produce a map of the walk. It also continuously logs the phone’s accelerometer data, giving information on the steepness, pitch/roll, and roughness of the track. The user can also add geo-tagged notes to the map, explaining possible obstacles or features of the track. Photos can be added to these markers for further detail.

## Application Vision

This App will provide the first objective grading system for wheelchair accessible bushwalks in Australia. The use of real world data from the accelerometers as a means of measuring not only the gradient and roll of the track, but also the roughness, will give a much better picture of the actual difficulty of the track in question than existing track grading, which relies on the topographical maps to get track gradient, and does not consider roll or roughness at all. Combining these features with GPS tracking and geo-tagged notes and pictures will for the first time give wheelchair-bound adventurers a clear picture of the difficulties faced on a graded track.

## Scope

This App is designed to be a data logging system only, and as such is intended to be used by a small group of wheelchair bound people in order to record the raw data of each walk and save it internally. The functionality of the app (at this stage) only covers the following:

* Walk Card display and simple walk database
* Sliding Navigation drawer
* Editing, Deleting, and Review of recorded walks
* Add new walk
* Google maps view
* Step and obstacle map markers
* Photo option for obstacle markers
* Logging of GPS, Accelerometer, and marker data
* Adding descriptions, rating, and thumbnail images to walks
* Minimal data processing for display purposes

At a future date I intend to also add the following functionality, and have included some of the necessary supporting framework for these features in the App:

* External accelerometer and odometer
* Export of log files to an external server
* Addition and logging of a heart rate monitor
* User editable survey platform information, that will be saved to the log file and provide context to the logged data
* Support for a more detailed topographical map tile set
* Support for offline map use
* Ability to resume a recording that was terminated

All final data processing will be done in a separate (PC) application to be developed at a later date. This is to allow for processing large data files from extended walks that require more processing power and screen real-estate than a phone allows.

## Document Version History

V1.0 – Initial project specification

V1.1 – Scope changes, functionality and features edited

V1.2 – Application structure and data structure

V1.3 – Edited functionality, APIs

V1.4 – Actual App screenshots and component overview

# Functionality Overview

The App boots to the Start Screen, displaying a card list with all the previously recorded walks that are stored in the database. Each walk can be edited by opening the edit dialog, and this gives the option to make and save changes, as well as review the log file or delete the walk.

A new walk recording can be started from either a button in the Action Bar, or from the Navigation Drawer. A new activity id=s started where the user enters the walk name and locality. Both of these fields must be completed and are used to generate the filename for the log file. Additional data on the survey platform and additional sensors can be entered here, but is currently not required for recording. Once the data has been entered, the save button advances the user to the recording activity

Recording to the log file is done in a background thread by the LogWriter service, with data collection from the GPS and the Accelerometers happening in their own services. The data is passed via intent broadcast to the LogWriter, and recorded into the log file. Data is also received by the interface, and displayed in either the Raw Data, Map, or Stats tab.

The Raw Data Tab allows the user to see the raw data from the GPS and Accelerometers as it is recorded.

The Map Tab allows the user to see their location and their path taken on the map, as well as add obstacle markers (with or without photos attached) and step markers to the map. Display of markers and track can be toggled on or off from buttons at the top of the map.

The Stats tab shows filtered data and basic statistics related to the log file. From this tab the user can also chose to discard the recording, which terminates the recording , deletes all data and returns to the Start Screen, or finalise the walk, which terminates the recording and passes the data on to the Finalise Walk Activity.

The Finalise Walk Activity allows the User to add a rating, description, and thumbnail image to the recorded walk, and finalises the log file with this added information. The walk is then passed back to the Start Screen Activity, where it is added to the database and displayed in the walk card list.

## Feature Summary

### Walk Card display and simple walk database

A RecyclerView showing each walk in the database with basic info and a thumbnail image

### Editing, Deleting, and Review of recorded walks

An edit dialog that allows the user to change the basic information stored in the database for each walk, and allows reviewing the log file and filtering it via data type. The user can either save or discard any changes, and delete the walk if the permission is set in the preferences.

### Add new walk

The user can begin recording a new walk, giving it a name and locality. The walk is then recorded in a log file (.txt) named using the name and locality.

### Google maps view

The GPS location of the user during recording is shown in a custom Google Map Fragment, and the path of the user is shown in red.

### Step and obstacle map markers

The user can add markers to the Map at their current position: either a Step marker showing the step height, or an Obstacle marker with a type and optional photo

### Photo option for obstacle markers

The user can choose to take a photo of an obstacle and have its path stored in the log file with the marker tag for later reference

### Logging of GPS, Accelerometer, and marker data

All data generated by the sensors (GPS and Accelerometer) as well as all user added markers are recorded to the log file with a corresponding Log Tag. When viewing the log later, this data can easily be filtered via tag.

### Adding descriptions, rating, and thumbnail images to walks

At the end of the walk. The User must finalise the recording by giving the walk a rating (based on how nice they thought the walk was), a grade (based on how difficult they thought it was), a brief description (this can be typed, or recorded using speech-to-text) and choosing a thumbnail image from the gallery.

### Minimal data processing for display purposes

Some minimal data processing is done in the background processes to calculate the average and max Pitch and Roll of the device, and the distance travelled. This data is for display purposes and is not recorded, as it can be better re-calculated from the data during analysis.

## Application Flow Diagram

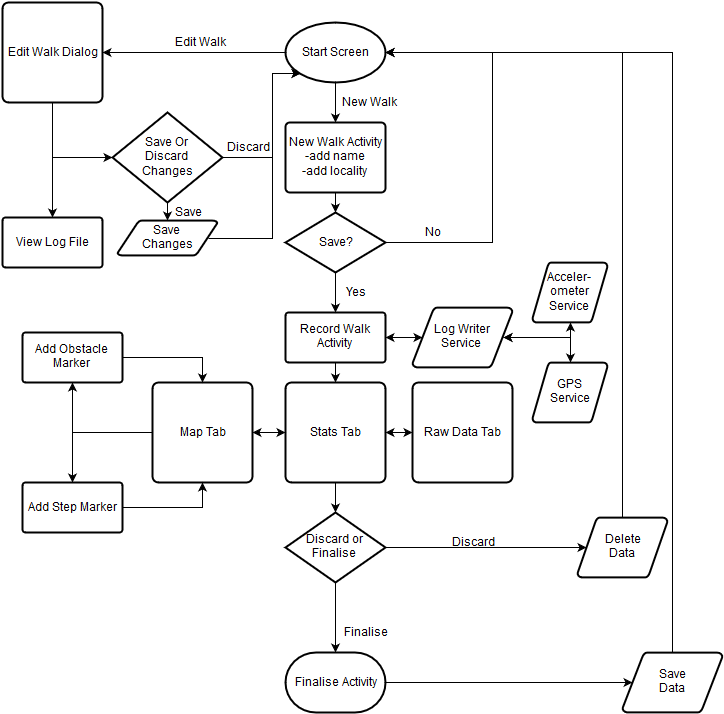
****

Figure 1 - Application Flow Diagram

# Design

## Overall Components overview

The main components of the Application are the Viewing and Editing activities, and the Recording activities. The StartScreenActivity acts as the home screen for the App, and provides information on walk already recorded, with the option to edit and view the walks in finer detail. From here the user can also change preferences (from the settings menu) which affect the whole App. Once a recording is started, these preferences are locked, and cannot be changed until the recording is finalised or discarded. This is to ensure that the log file data remains consistent throughout the recording.

The Recording activity is responsible for displaying the UI tabs, as well as starting and terminating the services that collect data and record the log file. This Activity contains a partial Wake-Lock, meaning that the activity and its services continue to run even when the phone goes to sleep, either by timeout or by pressing the power button. When the phone powers back on, the recording activity will show instead of the lock screen, and the user will be able to continue recording without unlocking the phone. Once the recording is finished, either by discarding or finalising, the user will need to unlock the phone before continuing. Once the recording is finalised, the log file and any associated images are saved to the external storage directory of the phone, and the walk is added to the database.

## Mock-ups

|  |  |
| --- | --- |
| Figure 2 - Start Screen | Figure 3 - Navigation Drawer |
| Figure 4 - Settings | Figure 5 - New walk Prompt |
| Figure 6 - New Walk Activity | Figure 7 - Record Walk Raw Data Tab |
| Figure 8 - Record Walk Map Tab | Figure 9 - Record Walk Add Step Dialog |
| Figure 10 - Record Walk Add Obstacle Dialog | Figure 11 - Map Now Showing Markers |
| Figure 12 - Record Walk Stats Tab | Figure 13 - Finalise Walk Warning |
| Figure 14 - Finalise Walk Activity | Figure 15 - Walk Added To Start Page |
| Figure 16 - Edit Walk Dialog | Figure 17 - View Log Dialog |
| Figure 18 - Filtered Log View | Figure 19 - Delete Walk Prompt |

## Used Application Programming Interfaces

### Google Maps API

Used to access and display the online maps, and show the track and marker objects

### Speech to Text API

Used for entering the walk description if desired, as an alternative to typing out a long description.

### GPS Services

Used for location tracking and logging

### Sensor Services

Used for obtaining the raw Accelerometer values for logging

### Gallery and Camera

Used for choosing the Thumbnail image when finalising a walk, and for taking photos of obstacles during recording.

# Application Structure

## Fragments

### AddObstacleDialogFragment

Triggered by the CustomMapFragment

Shows a Dialog allowing the user to choose from a dropdown the type of obstacle encountered and optionally take a photo of the obstacle. Returns the type and photo via intent to the CustomMapFragment

### AddStepDialogFragment

Triggered by the CustomMapFragment

Shows a Dialog allowing the user to select the height of the step. Returns the height via intent to the CustomMapFragment

### CustomWalkMapFragment

A custom implementation of a GoogleMap SupportMapFragment that is instantiated at runtime and provides functionality for adding custom markers and a path.

### EditWalkDialogFragment

A Dialog that is passed A walk object and allows the user to view and edit the Walk information.

### RawDataFragment

A tabbed view that displays raw data as it comes in from the phones sensors

### SearchWalksDialogFragment

A Dialog that allows the user to search the walks database. The search takes a string from the user and looks for partial matches in the name and locality columns of the database.

### StatsFragment

A tabbed view that shows Statistics regarding the recorded data, such as average and maximum angle of the sensors.

### WalkMapFragment

A tabbed fragment that holds a CustomMapFragment (instantiated at runtime) and several controls to interact with it: add markers, toggle markers/path on or off.

### LogFileViewerDialog

Displays a Dialog containing the text of the log file. The log file is loaded in an asyncTask line by line, with each line being checked for its tag and given a colour according to the data type. A group of checkboxes in the dialog allow the user to apply filters to the data, restarting the asyncTask each time a filter is changed.

## Activities

### StartScreenActivity

The starting page of the App. Shows a Recyclerview Card list containing all the walks in the database. These walks can be searched via a button in the ActionBar or the Navigation Drawer. The App returns to this page after recording and finalising a new walk, with the new walk added to the database and cardlist.

### NewWalkActivity

This page prompts the user to enter information required to start a new recording. Checks for missing data and duplicate log files, avoiding overwriting existing walks.

### RecordWalkActivity

This activity is the parent activity of the tabbed fragment views. It starts the LogWriter service and shuts it down when finished. Holds the phone in a partial wake lock during recording to avoid being interrupted, and will show before the lockscreen if the display is turned back on after being turned off. The recording continues to run even when the screen is off.

### FinaliseWalkActivity

Prompts the user to enter a rating, Grade, Description, and choose a thumbnail image for the walk, and saves these to the log file before returning the walk to the StartScreenActivity.

### SettingsActivity

Allows the user to set up preferences that affect the user experience.

## Services

### AccelerometerManager

Sets up and reads the sensors periodically, passing the data on via intent to the LogWriter

### GpsTracker

Sets up and listens for GPS updates, passing the data on via intent to the LogWriter and CustomMapFragment.

### LogWriter

Sets up the AccelerometerManager and GpsTracker, and listens for data to be recorded. Also listens for updates from the UI such as markers to be added to the log file.

## Models

### AccelerometerData

Holds the XYZ raw sensor data in a parcelable format.

### Walk

Holds the walk data in a parcelable format.

### Platform

Holds data on the type of platform used for surveying. Currently not used.

### WalkDbConstants

Holds the constants for interacting with the database (table/column names etc.)

### Constants

Holds all constants used by the App, sorted by function.

## Controllers

### NavDrawerAdapter

Adapter for the Navigation drawer on the StartScreenActivity

### WalkCardViewAdapter

Adapter for the Walk Card RecyclerView on the StartScreenActivity

### WalkCardViewHolder

ViewHolder for the Card View. Also handles the buttons on each card.

### WalksDatabaseHelper

The Helper class for setting up and accessing the Walks database. All SQLite code resides here.

## Utilities

### Utils

Contains common methods used by many different classes in the App.

# Data Structure

As this project stores all logging data in the log file itself, the only database used is a single table SQLite database used to list the walks that have been recorded. Each row of the Walks table contains the information for a single walk. The columns of the Walks table are as follows:

* \_id – auto-incremented int
* Name – String name of the walk
* Date\_surveyed – Long UTC time the recording was started
* Length – length of the recorded walk in meters
* Rating – 0-5 float star rating of the walk
* Grade – 1-5 int grade of difficulty
* Description – String user description of the walk
* Locality – string area in which the walk is located
* Image\_file – String absolute path of the walk thumbnail
* Log\_file– String absolute path of the walk log file

All this information (except the id and log file path) is also contained in the header and footer of the log file.

# Testing

|  |  |  |
| --- | --- | --- |
| **Pre-Condition** | **Post-Condition** | **Pass/Fail** |
| The user searches for a walk with a partial string that is found in several walks | All walks containing the partial string are shown | Pass |
| The user searches for a string that is not contained in any walks | The Start screen shows all walks, with a toast saying the search returned no results | Pass |
| The User makes changes to a walk in the edit dialog, and then discards them | The walk in the database is unchanged | Pass |
| The User makes changes to a walk in the edit dialog, and then saves | The changes are saved to the database, and the card view is updated to reflect the changes | Pass |
| The user starts a new walk recording with a new name and locality | The recording starts successfully with the name and locality in the log file name | Pass |
| The user starts a new walk recording with no name and locality | The UI prompts for the fields to be populated | Pass |
| The user starts a new walk recording with name and locality that have already been recorded | The UI warns that a recording for that walk already exists, and prompts for a new entry | Pass |
| The user clicks the add Step Tag and selects cancel in the dialog | No marker is added | Pass |
| The user clicks the add Obstacle Tag and selects cancel in the dialog | No marker is added | Pass |
| The user clicks the add Step Tag and selects a height, then clicks save | A marker is added, with the selected height as subtext, and the info is added to the log file | Pass |
| The user clicks the add Obstacle Tag and selects a Type, then clicks save | A marker is added, with the selected type as subtext, and the info is added to the log file | Pass |
| The user clicks the add Obstacle Tag and selects a Type, then takes a photo and clicks save | The marker is created as above, and the file path of the photo is saved to the log file along with the marker info | Pass |
| While recording, the user presses the phone power button to put the phone into sleep mode | The display turns off, but the recording continues, as seen in the debug console | Pass |
| The user then turns the display back on after sleeping | The recording Activity is shown in place of the lock screen, and the data from when the phone was asleep is visible on the map and in the log file | Pass |
| The user selects the discard recording button | The recorded log file is deleted (after confirmation) and no walk is added to the database. All services are correctly terminated | Pass |
| The user selects the finalise walk button | The recorded log file is saved (after confirmation) the walk is passed on to the FinaliseWalkActivity. All services are correctly terminated | Pass |
| The user Finalises the walk without entering a rating/description/grade/thumbnail | The UI prompts for those fields to be populated | Pass |
| The user Finalises the walk after filling out all the fields | The final data is appended to the log file, and the walk is returned to the StartScreenActivity, where it is added to the database and displayed on the card view. | Pass |

# Conclusion

I found this to be a very rewarding project to develop, and a great learning experience. Having to research and implement so many new features kept the project exciting, and knowing that it may actually be of benefit to my friend and to wider society provided me with a lot of motivation to work hard on the project. I still see this App as a prototype, with much further development required before I can call it a finished product. This is due solely to time constraints imposed by the subject and by conflicts with work and other classes. I have so many ideas for extra features I want to add into this project that it will probably take me months more to finish it. I also intend to develop a PC application for doing the data analysis (hopefully taking advantage of cloud computing) and a second App that anyone can download to use the recorded data and go on the walks we have graded.

The App is not as polished as I would like it to be, partially because of my complete lack of artistic talent, but mainly because I much prefer implementing new features to polishing existing ones. This is something I think I need to work on, and I need to adopt a zero-error policy (no new code until all known bugs are fixed) in future android projects to avoid wasted time on workarounds.

Everything I tried to implement on the project worked (eventually), though not always in the way I expected or planned, and I guess I have to thank Stack Overflow for many of the tutorials and code snippets that put me on the right track.

I am happy with how the functionality of the App has turned out, and hoe to get some feedback on possible changes and improvements that I can incorporate into the next version.

# References

**Stack Overflow** - <http://stackoverflow.com/questions/> - Code snippets and coding techniques

**Icons 4 Android** - <http://icons4android.com/> - Android Icons

**Tutorials Point** - <http://www.tutorialspoint.com/android/> - Code snippets and tutorials

**Lynda** - <http://www.lynda.com/Android-training-tutorials> - Code snippets and tutorials

**Google Developers** - <http://developer.android.com/training> - Code snippets and tutorials

**Android Asset Studio** - <https://romannurik.github.io/AndroidAssetStudio/index.html> - Custom generated icon sets

**Android Material Design** - <https://developer.android.com/design/material/index.html> -Design guidelines and colour swatches